European Journal of Operational Research 40 (1989) 90–98
North-Holland

Theory and Methodology

A new integer programming formulation for the permutation flowshop problem

A.M. FRIEZE *
Department of Computer Science and Statistics, Queen Mary College, London University, England, UK

J. YADEGAR **
DAP Support Unit, Queen Mary College, London University, England, UK

Abstract: We describe a new integer programming formulation for the permutation flowshop problem in which the objective is to minimise the makespan. This formulation can have an exponential number of constraints, but its linear programming relaxation can be solved by a novel (row generation) algorithm in polynomial time. We present some computational experience.

Keywords: Scheduling, integer programming, assignment, heuristic, graphs

1. Introduction

In this paper, we discuss a new integer programming formulation for the problem of minimising *makespan* in a *permutation flowshop.*

The usual assumptions about the permutation flowshop problem will be adopted. Thus, assume that we have n jobs $J_1, J_2, \ldots, J_n$ which have to be processed on m machines $M_1, M_2, \ldots, M_m$ in this order, and we let p_{ij} be the processing time for J_i on M_j. At any time, each machine can process at most one job and each job can be processed on at most one machine. Once the processing of a job on a machine has started, it must be completed without interruption. A feasible schedule can be represented by a permutation σ of $\{1, 2, \ldots, n\}$. This means that each machine processes the jobs in the order $J_{\sigma(1)}, J_{\sigma(2)}, \ldots, J_{\sigma(n)}$.

We let $F(\sigma)$ denote the elapsed time between the start of $J_{\sigma(1)}$ on M_1 and the completion of $J_{\sigma(n)}$ on M_m. This is referred to as the *makespan* or the flow-time of sequence σ. Then the problem is to

$$\underset{\sigma \in S_n}{\text{minimise}} \quad F(\sigma), \tag{1}$$

where $S_n = \{\sigma : \sigma \text{ is a permutation of } \{1, 2, \ldots, n\}\}$.

For $m = 2$, the optimal schedule can be obtained in $O(n \log n)$ steps by Johnson's algorithm [16]. However, for $m = 3$, the problem is known to be NP-Hard—see Garey and Johnson [9], Garey, Johnson and Sethi [10], or Lenstra, Rinnooy Kan and Brucker [21]—and it has an extensive literature. For a review on the flowshop problem, we refer the reader to Baker [1,2], Bellman, Esogbue and Nabeshima [3], Bestwick and Hastings [4], Brown and Lomnicki [5], Campbell, Dudek and Smith [6], Conway, Maxwell and Miller [7],

* Present address: Department of Mathematics, Carnegie-Mellon University, Pittsburgh, USA.

** This research was funded in part by a University of London Studentship. Present address: Active Memory Technology, Inc., Irvine, CA 92714, USA.

Received February 1987; April 1988

Graham et al. [13], Ignall and Schrage [15], Lageweg, Lenstra and Rinnooy Kan [18], Lenstra [20], Lomnicki [22], McMahon and Burton [23], Potts [24], Rinnooy Kan [25], and Szwarc [27].

In Section 2, we describe our integer programming formulation for the makespan problem and in Section 3, we present an algorithm for solving the linear programming relaxation of the integer program, giving a lower bound. This bound is then compared with known bounds. Finally, computational experience is reported in Section 4 which is followed by some concluding remarks in Section 5.

2. An integer programming formulation for the makespan problem

It is well known that for a given σ, we can compute $F(\sigma)$ as the length of the longest path in a certain acyclic digraph $G_\sigma = (N, A)$—see Figure 1—where the set of nodes N is given by

$$N = \{s, f\} \cup \{[k, l] : k = 1, 2, \ldots, n, \quad l = 1, 2, \ldots, m\},$$

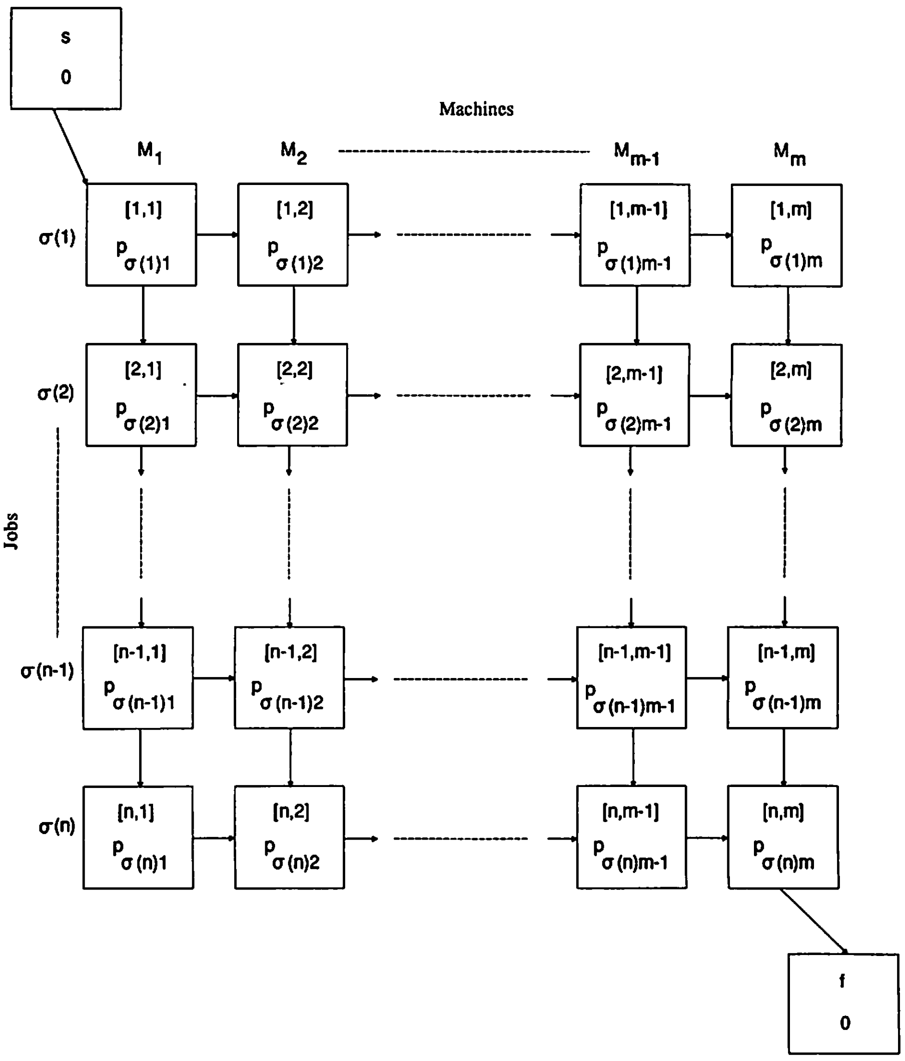

Figure 1. $G_\sigma = (N, A)$

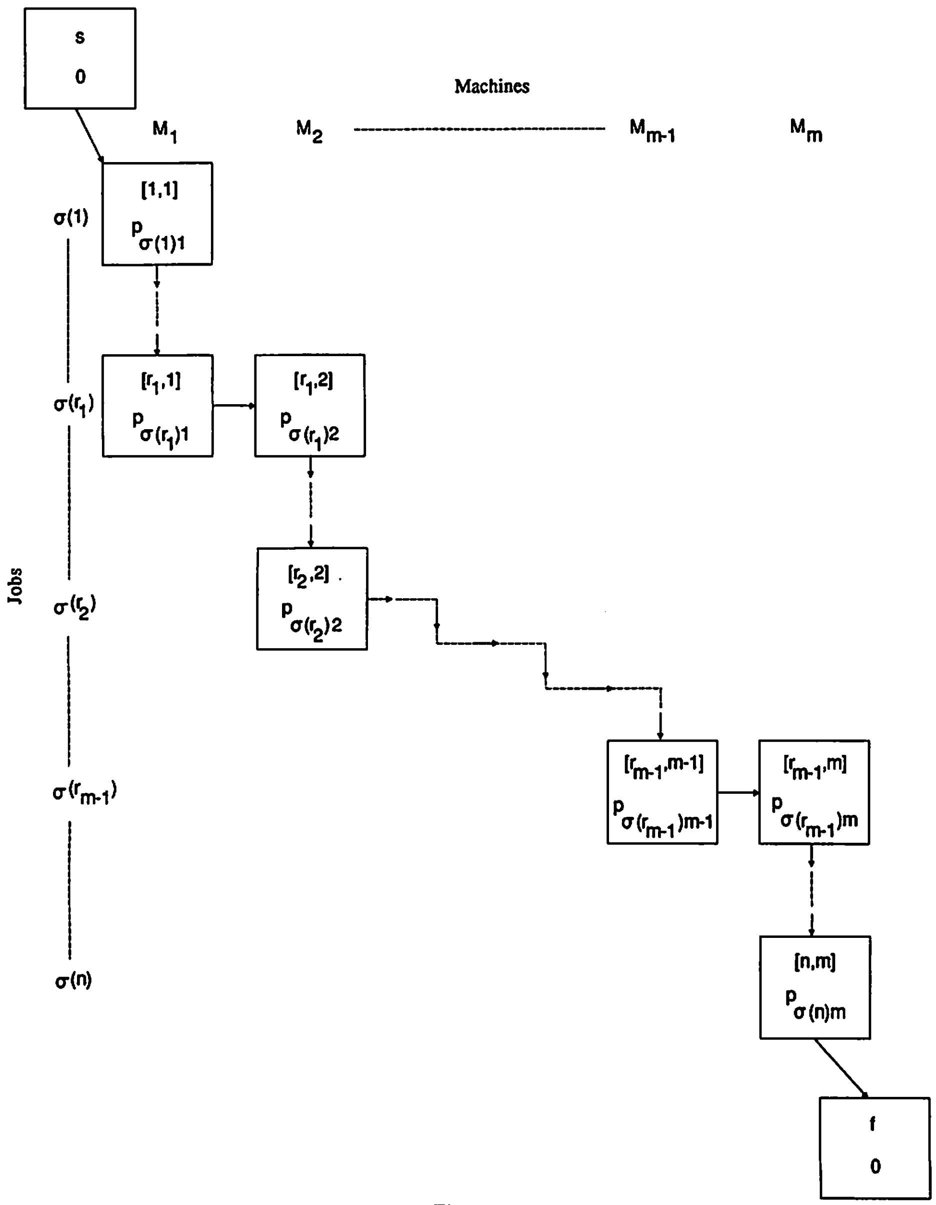

Figure 2

and

$$A=\{(s,[1,1]),([n,m],f)\}$$
$$\cup\{([k,l],[k+1,l]):k=1,2,\ldots,n-1,\quad l=1,2,\ldots,m\}$$
$$\cup\{([k,l],[k,l+1]):k=1,2,\ldots,n,\quad l=1,2,\ldots,m-1\}.$$

Note that N and A are independent of σ. However, each arc u has a length $l(u)$ associated with it. That is,

$$l(s,[1,1])=0$$

and

$$l(u)=p_{\sigma(i)j}\quad \text{if arc } u \text{ has tail } [i,j] \text{ for all } i=1,\ldots,n \text{ and } j=1,\ldots,m.$$

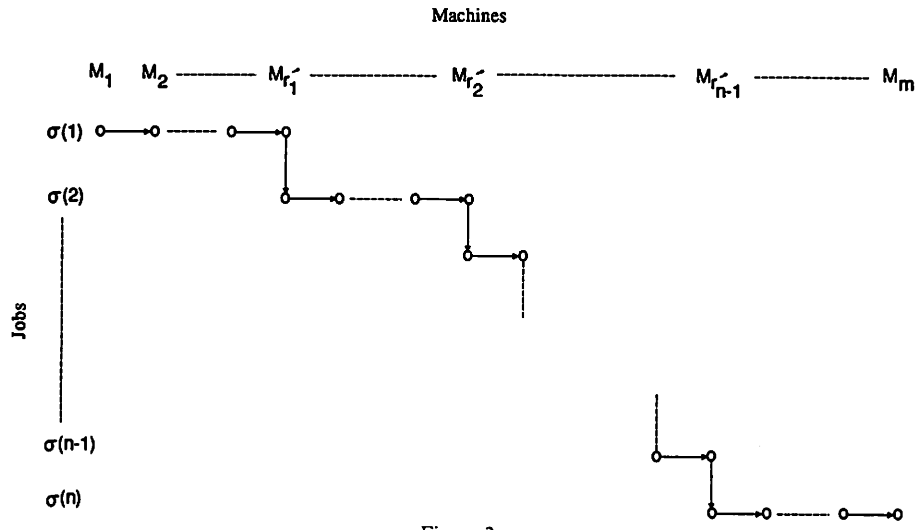

Figure 3

Then, as shown in reference [3, pp. 141–145], we have

$F(\sigma)$ = the length of the longest path from s to f in G_σ.

Now let T be the set of paths from s to f in G_σ. We can represent a path $\tau \in T$ by a sequence of integers $1 \leqslant r_1 \leqslant r_2 \leqslant \cdots \leqslant r_{m-1} \leqslant n$—see Figure 2—indicating the positions of the 'horizontal' sections of τ. Thus τ is the set of arcs

$$A_\tau = \{(s, [1, 1])\} \cup \bigcup_{k=1}^{m} \{\{([i, k], [i+1, k]) : r_{k-1} \leqslant i \leqslant r_k\} \cup \{([r_k, k], [r_k, k+1])\}\},$$

where $r_0 = 1$, $r_m = n$ and $[r_m, m+1]$ denotes f.

We let

$\Gamma(\tau, \sigma)$ = the length of the path τ in G_σ.

Then Problem (1) becomes:

$$\underset{\sigma \in S_n}{\text{minimise}} \quad \underset{\tau \in T}{\text{maximum}}\, \Gamma(\tau, \sigma). \tag{2}$$

We now introduce an $n \times n$ permutation matrix $X^\sigma = \| x_{ij}^\sigma \|$ for each $\sigma \in S_n$, where

$$x_{ij}^\sigma = \begin{cases} 1 & \text{if } i = \sigma(j), \\ 0 & \text{otherwise.} \end{cases}$$

Note that the mapping $\sigma \to X^\sigma$ is 1–1.

We now express $\Gamma(\tau, \sigma)$ in terms of X^σ.

Now for $\tau = (r_1, r_2, \ldots, r_{m-1})$, we can see from the definition of A_τ that

$$\Gamma(\tau, \sigma) = \sum_{k=1}^{m} \sum_{j=r_{k-1}}^{r_k} p_{\sigma(j)k} \tag{3}$$

$$= \sum_{k=1}^{m} \sum_{j=r_{k-1}}^{r_k} \sum_{i=1}^{n} p_{\sigma(i)k} x_{\sigma(i)j}^\sigma$$

$$= \sum_{k=1}^{m} \sum_{j=r_{k-1}}^{r_k} \sum_{i=1}^{n} p_{ik} x_{ij}^\sigma \tag{4}$$

$$= \sum_{i=1}^{n} \sum_{k=1}^{m} \sum_{j=r_{k-1}}^{r_k} p_{ik} x_{ij}^\sigma$$

$$= \sum_{i=1}^{n} \sum_{j=1}^{n} p_{ij}^\tau x_{ij}^\sigma,$$

where

$$p_{ij}^\tau = \sum_{k : r_{k-1} \leqslant j \leqslant r_k} p_{ik}, \quad i, j = 1, 2, \ldots, n. \tag{5}$$

Thus, if P^τ is the $n \times n$ matrix $\| p_{ij}^\tau \|$, then Problem (2) can be written as the following integer program:

minimise ξ

subject to

$$\xi - P^\tau X \geqslant 0, \quad \tau \in T, \tag{6a}$$

$$\sum_{i=1}^{n} x_{ij} = 1, \quad j = 1, 2, \ldots, n, \tag{6b}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \quad i = 1, 2, \ldots, n, \tag{6c}$$

$$x_{ij} = 0 \text{ or } 1, \quad i, j = 1, 2, \ldots, n, \tag{6d}$$

where $P^\tau X = \Sigma_{i=1}^{n} \Sigma_{j=1}^{n} p_{ij}^\tau x_{ij}$ for $\tau \in T$.

Note that as

$$|T| = \binom{n+m-2}{m-1},$$

this formulation can have an exponential number of constraints.

It is worth noting that a path $\tau \in T$ can also be represented by an $(n-1)$-tuple $1 \leqslant r_1' \leqslant r_2' \leqslant \cdots \leqslant r_{n-1}' \leqslant m$ of integers—see Figure 3—if we consider individual jobs on various machines rather than considering various jobs on individual machines. In this case, the expression for $\Gamma(\tau, \sigma)$ becomes,

$$\begin{aligned}\Gamma(\tau, \sigma) &= \sum_{i=1}^{n} \sum_{k=r_{i-1}'}^{r_i'} p_{\sigma(i)k}, \quad r_0' = 1, \ r_n' = m \\ &= \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=r_{j-1}'}^{r_j'} p_{\sigma(i)k} x_{\sigma(i)j}^\sigma \\ &= \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=r_{j-1}'}^{r_j'} p_{ik} x_{ij}^\sigma \\ &= \sum_{i=1}^{n} \sum_{j=1}^{n} \pi_{ij}^\tau x_{ij}^\sigma,\end{aligned}$$

where

$$\pi_{ij}^\tau = \sum_{k=r_{j-1}'}^{r_j'} p_{ik}, \quad i, j = 1, 2, \ldots, n. \tag{7}$$

It is not difficult to see that the expression given for π_{ij}^τ in (7) is the same as that given for p_{ij}^τ in (5), and thus one obtains the same integer programming formulation as that of (6).

We next show how to solve the usual linear programming relaxation of the integer program (6). This, generally, yields a good lower bound on the optimal value of the makespan problem.

3. Lower bounds

Let LP denote the linear programming relaxation of (6)—i.e. replace (6d) by $x_{ij} \geqslant 0$. Because of the large number of constraints, it is clear that the only feasible approach to solving LP is by some row generation scheme. One possibility is described next. For $T^* \subseteq T$ let LP(T^*) denote the linear program obtained from LP by replacing (6a) by

$$\xi - P^\tau X \geqslant 0, \quad \tau \in T^*. \tag{8}$$

Given T^* we solve the LP relaxation LP(T^*), obtaining ξ^*, X^*. We then check to see if ξ^*, X^* is feasible for LP. If it is we are done, otherwise we add a violated constraint and throw away slack constraints, and repeat. Formally:

3.1. *Algorithm* BOUND

```
begin
   choose an initial set of paths $T^* \subseteq T$;
   bub := $\infty$ — best (smallest) upper bound on the
                 optimal value of LP computed so
                 far;
   blb := $-\infty$ — best (largest) lower bound on the
                 optimal value of LP computed so
                 far;
   repeat
A:    let $\xi^*$, $X^*$ be an optimal solution to
      LP($T^*$);
B:    let $\mu = P^{\tau^*}X^* = \max_{\tau \in T} P^\tau X^*$;
      bub := min (bub, $\mu$);
      if $\xi^* \geqslant$ bub then terminate
      {remark: $\xi^*$, $X^*$ is optimal}
      else if $\xi^*$ = blb then $T^* := T^* \cup \{\tau^*\}$
         else {remark: $\xi^* >$ blb}
         begin
            blb := $\xi^*$;
C:          $T^* := (T^* \cup \{t^*\})$
                 $- \{\tau \in T^* : \xi^* > P^\tau X^*\}$
         end
      until termination
   end
```

Note that the sequence of minima ξ^*'s obtained by relaxation in Statement A is monotone non-decreasing.

Now, it is well known—see Lasdon [19]—that Algorithm BOUND will solve the linear programming problem LP.

ξ^*, X^* in Statement A can be found using the simplex algorithm as we can expect $|T^*|$ to be relatively small ($|T^*| \leqslant n^2 - 2n + 2$ if we only keep the minimum number of tight constraints needed to define ξ^*, X^*).

We now have to explain how to execute Statement B efficiently. If X^* were an integer matrix, i.e. $X^* = X^\sigma$ for some $\sigma \in S_n$, then we simply find the longest path in G_σ. For X^* non-integer we can do the same, i.e. define a digraph G_{X^*} and find its longest path.

Now, G_{X^*} has the same nodes N and arcs A as before. Arc lengths are as defined below.

From (4) we have

$$P^\tau X^* = \sum_{k=1}^{m} \sum_{j=r_{k-1}}^{r_k} \sum_{i=1}^{n} p_{ik} x_{ij}^* = \sum_{k=1}^{m} \sum_{j=r_{k-1}}^{r_k} p_{jk}^*, \tag{9}$$

where

$$p_{jk}^* = \sum_{i=1}^{n} p_{ik} x_{ij}^* \quad \text{for } j = 1, 2, \ldots, n, \; k = 1, 2, \ldots, m.$$

Comparing (9) and (3), we see that $P^\tau X^*$ is indeed the length of the path τ if arcs having tail $[j, k]$ have length p_{jk}^* for all j, k. Using this definition of arc length, we see that the maximisation problem in Statement B is a longest path problem and is solvable in O(mn) time. (In the terminology of Grötschel, Lovâsz and Schrijver [14], the separation problem for the polyhedral feasible region of the linear programming problem LP is solvable in polynomial time. Hence, LP is solvable in polynomial time using their variant of Khachian's algorithm [17].)

We note, using [28], that

$$\xi^* = \text{minimum}\Big(\sum_{\tau \in T^*} \lambda_\tau^* P^\tau \Big) X \tag{10}$$

subject to (6b), (6c) and (6d),

where λ^* is the optimum dual solution to LP(T^*). The solution to the assignment problem (10) was used to provide heuristic solutions to the flowshop problem.

3.2. Comparison with other lower bounds

It is easy to see that

$$\text{LB} = \max_{\tau \in T} \min_{\sigma \in S} \Gamma(\tau, \sigma) \tag{11a}$$

$$\leqslant \xi^* \leqslant \min_{\sigma \in S} \max_{\tau \in T} \Gamma(\tau, \sigma) = \hat{\xi}, \tag{11b}$$

where $\hat{\xi}$ and ξ^* are the optimal values to the integer program (6) and its linear relaxation LP respectively.

It is worth noting that for a fixed path $\tau \in T$, the problem $\min_{\sigma \in S} \Gamma(\tau, \sigma)$ ($= \min_{X \in \Lambda} P^\tau X$, where Λ is the set of 0–1 X's, satisfying constraints (6b), (6c), (6d)) is an assignment problem. One can actually give an interpretation for this assignment problem: namely, shuffle the rows (jobs) in the digraph G_σ such that the length of path τ is minimised.

We will now show that there exists a subset $T_{\text{NP}} = \{\tau_1, \tau_2, \ldots, \tau_m\}$ of T which in place of T in (11a) reduces the lower bound LB to the machine-based bound obtained by Brown and Lomnicki [5] and Ignall and Schrage [15]. For this purpose, let us first define the elements of T_{NP} by using the $(m-1)$-tuple representation of a path. That is,

$$T_l = (1, \ldots, 1, n, \ldots, n) \quad \text{for } l = 1, \ldots, m-1$$

and

$$T_m = (1, 1, \ldots, 1),$$

where the first n in τ_l occurs in the l-th place.

Therefore, the l-th path of T_{NP} in G_σ of Figure 1 is the horizontal path from node [1, 1] to node [1, l], then vertically down from node [1, l] to node [n, l], followed by the horizontal path from node [n, l] to node [n, m].

We next define:

$$\text{LB}_l = \min_{\sigma \in S} \Gamma(\tau_l, \sigma) \quad \text{for } l = 1, \ldots, m$$

and

$$\overline{\text{LB}} = \max_{1 \leqslant l \leqslant m} \text{LB}_l.$$

Now, using (3), it is easy to see that

$$\begin{aligned} \text{LB}_l &= \min_{\sigma \in S} \Big\{ (1 - \delta_{1l}) \sum_{k=1}^{l-1} p_{\sigma(1)k} + \sum_{j=1}^{n} p_{\sigma(j)l} + (1 - \delta_{ml}) \sum_{k=l+1}^{m} p_{\sigma(n)k} \Big\} \\ &= \sum_{j=1}^{n} p_{jl} + \min_{\substack{1 \leqslant \sigma(1), \sigma(n) \leqslant n \\ \sigma(1) \neq \sigma(n)}} \Big\{ (1 - \delta_{1l}) \sum_{k=1}^{l-1} p_{\sigma(1)k} + (1 - \delta_{ml}) \sum_{k=l+1}^{m} p_{\sigma(n)k} \Big\}, \end{aligned}$$

where δ_{kl} is the Kronecker delta.

It is now clear that $\overline{\text{LB}}$ is indeed the machine-based lower bound produced by Brown and Lomnicki [5] and also by Ignall and Schrage [15]. Furthermore, since $\overline{\text{LB}} \leqslant \text{LB}$, it follows that the bounds of [5] and [15] are weaker than the bound obtained by Algorithm BOUND.

The bound of Bestwick and Hastings [4], the job-based bound introduced by McMahon and Burton [23], and the two-machine bound proposed by Lageweg et al. [18] and Potts [24] are incomparable with our bound.

4. Computational experience

In each iteration of Algorithm BOUND, we also compute an upper bound on $\hat{\xi}$, that is, on the value of an optimal permutation schedule for the flowshop problem in (6). This is done as follows:

Let λ^* be an optimal multiplier vector (dual solution) associated with constraints (8) in the linear program LP(T^*), which is readily available from the optimal vector of shadow prices for LP(T^*). Now, from Geoffrion [11], the vector λ^* is also optimal for the dual problem

$$\max_{\lambda : \Sigma_{\tau \in T} \lambda_\tau = 1 \text{ and } \lambda_\tau \geqslant 0} \rho(\lambda) \quad ,$$

where

$$\varphi(\lambda) = \min_{X \in A} \left(\sum_{\tau \in T} \lambda_\tau P^\tau \right) X.$$

Thus λ^* can be used in (10) to achieve (hopefully) a good permutation schedule X^σ satisfying (6b), (6c) and (6d). We then use Statement B in Algorithm BOUND (i.e. compute the longest path in G_σ) to find a feasible solution for (6).

No special method was employed to solve the linear program LP(T^*). We had a program available, based on the simplex algorithm for solving a general linear program with bounded variables, and so we used it. A primal–dual algorithm based on that of Ford and Fulkerson [8] was used to solve the weighted assignment problem (10).

We have carried out some computational experiments to try to evaluate the strength of the proposed bound. The results of these experiments are given in Table 1. The algorithm was coded in FORTRAN IV and run on an ICL 2980 computer.

Explanation of Table 1

P	= Source of problems: Problem 1 is from Ignall and Schrage [15]. Problems 2–3 are from Brown and Lomnicki [5]. Problems 4–9 are from Giglio and Wagner [12]. Problem 10 is from Smith and Dudek [26]. Problems 11–15 are randomly generated. The processing times are integers selected from the interval [10, 40] for problems 11–12, and from the interval [0, 10] for problems 13–15.
n	= Number of jobs.
m	= Number of machines

Table 1

P	n	m	T^* [a]	$\lceil \xi^* \rceil$	ub	t	MB	JB	TMB	bv
1	10	3	15	61	66	6.55	59	61	65	66 [b]
2	6	3	6	99	114	0.70	99	107	107	109 [b]
3	6	5	20	85	97	4.79	75	90	90	97 [b]
4	6	3	4	55	59	0.63	53	54	54	57 [b]
5	6	3	1	64	64	0.54	64	61	64	64 [b]
6	6	3	5	69	80	0.77	69	69	69	69 [b]
7	6	3	1	63	63	0.57	63	49	63	63 [b]
8	6	3	8	65	70	1.03	63	68	68	68 [b]
9	6	3	8	69	81	1.04	67	75	75	76 [b]
10	5	3	2	1051	1086	0.62	1051	1011	1051	1078 [b]
11	5	3	7	179	202	0.73	161	202	202	202 [b]
12	7	3	7	221	241	1.14	214	240	240	241
13	7	4	2	35	38	0.63	34	22	35	38
14	10	3	8	66	68	1.78	65	65	66	68
15	10	4	10	48	62	1.15	47	44	48	62

[a] The computer program used did not implement Statement C in Algorithm BOUND. Thus, the true number of paths needed to solve LP will be less than or equal those indicated in Table 1.
[b] Indicates that this is known to be optimal.

T^*	= Number of paths used before reaching the optimal solution of LP.
ξ^*	= The optimal value of LP. Since the processing times are integers, it follows that $\lceil \xi^* \rceil$ is a valid lower bound.
ub	= The best solution value computed.
t	= The CPU time in seconds on an ICL 2980 computer.
MB	= The bound obtained using the machine-based bound.
JB	= The bound obtained using the job-based bound.
TMB	= The bound obtained using the two-machine bound.
bv	= The value of the best known solution to these problems.

5. Concluding remarks

We have constructed a new integer programming formulation of the flowshop problem. The lower bounds computed are either independent or superior to the well-known bounds. The most interesting aspect seems to us to be the way we have extended the use of the acyclic digraph G_o to deal with fractional solutions.

The computational results are not sensational but there is hope that the bound can be useful as an addition to the two machine bound, which has tended to dominate our bound in the tests. The computation time for the bound does not seem to be a problem, given that we did not use a very efficient version of the simplex algorithm in our tests. Note also that the number of tight constraints needed to define a basic solution is at most $n^2 - 2n + 2$, where n is the number of jobs and so is independent of the number of machines.

Acknowledgements

We thank Chris Potts for providing us with some useful information.

References

[1] Baker, K.R., *Introduction to sequencing and scheduling*, Wiley, New York, 1974.

[2] Baker, K.R., "A comparative study of flow-shop algorithms", *Operations Research* 23 (1975) 62–73.

[3] Bellman, R., A.O. Esogbue, and Nabeshima, I., *Mathematical Aspects of Scheduling and Applications*, Pergamon Press, Oxford, 1982.

[4] Bestwick, P.F., and Hastings, N.A.J., "A new bound for machine scheduling", *Operational Research Quarterly* 27, (1976) 479–487.

[5] Brown, A.P.G., and Lomnicki, Z.A., "Some applications of the branch-and-bound algorithm to the machine scheduling problem", *Operational Research Quarterly* 17 (1966) 173–186.

[6] Campbell, H.G., Dudek, R.A., and Smith, M.L., "A heuristic algorithm for the *n* job, *m* machine sequencing problem", *Management Science* 16 (1970) B630–B637.

[7] Conway, R.W., Maxwell, W.L., and Miller, L.W., *Theory of Scheduling*, Addison-Wesley, Reading, MA, 1967.

[8] Ford, L.R., and Fulkerson, D.R., *Flows in Networks*, Princeton University Press, Princeton, NJ, 1962.

[9] Garey, M.R., and Johnson, D.S., *Computers and Intractibility: A Guide to NP-Completeness*, Freeman and Company, San Francisco, CA, 1979.

[10] Garey, M.R., Johnson, D.S., and Sethi, R., "The complexity of flowshop and jobshop scheduling", *Mathematics of Operations Research* 1 (1976) 117–129.

[11] Geoffrion, A.M., "Lagrangean relaxation for integer programming", *Mathematical Programming Study* 2 (1974) 82–114.

[12] Giglio, R.J., and Wagner, H.M., "Approximate solutions to the three-machine scheduling problem", *Operations Research* 12 (1964) 305–324.

[13] Graham, R.L., Lawler, E.L., Lenstra, J.K., and Rinnooy Kan, A.H.G., "Optimization and approximation in deterministic sequencing and scheduling: A survey", *Annals of Discrete Mathematic* 5 (1979) 287–326.

[14] Grötschel, M., Lovász, L., and Schrijver, A., "The ellipsoid method and its consequences in combinatorial optimization", *Combinatorica* 1 (1981) 169–197.

[15] Ignall, E., and Schrage, L., "Application of the branch and bound technique to some flow-shop scheduling problems", *Operations Research* 13 (1965) 400–412.

[16] Johnson, S.M., "Optimal two- and three-stage production schedules with setup times included", *Naval Research Logistic Quarterly* 1 (1954) 61–68.

[17] Khachian, L.G., "A polynomial algorithm in linear programming", *Doklady Akademii Nauk USSR, Nova Seria* 244/5 (1979) 1093–1096; Translated in *Soviet Mathematics Doklady* 20 (1979) 191–194.

[18] Lageweg, B.J., Lenstra, J.K., and Rinnooy Kan, A.H.G., "A general bounding scheme for the permutation flowshop problem", *Operations Research* 26 (1978) 53–67.

[19] Lasdon, L.S., *Optimization Theory for Large Systems*, MacMillan, London, 1970.

[20] Lenstra, J.K., *Sequencing by Enumerative Methods*, Mathematical Centre Tract 69, Mathematisch Centrum, Amsterdam, 1977.

[21] Lenstra, J.K., Rinnooy Kan, A.H.G., and Brucker, P., "Complexity of machine scheduling problems", *Annals of Discrete Mathematics* 1 (1977) 343–362.

[22] Lomnicki, Z.A., "A branch-and-bound algorithm for the exact solution of the three-machine scheduling problem", *Operational Research Quarterly* 16 (1965) 89–100.

[23] McMahon, G.B., and Burton, P.G., "Flow-shop scheduling with the branch-and-bound method", *Operations Research* 15 (1967) 473–481.

[24] Potts, C.N., "An adaptive branching rule for the permutation flow-shop problem", *European Journal of Operational Research* 5 (1980) 19–25.

[25] Rinnooy Kan, A.H.G., *Machine Scheduling Problems: Classification, Complexity and Computations*, Nijhoff, The Hague, 1976.

[26] Smith, R.D., and Dudek, R.A., "A general algorithm for solution of the n-job, M-Machine sequencing problem of the flowshop", *Operations Research* 15 (1967) 71–82.

[27] Szwarc, W., "Elimination methods in the $m \times n$ sequencing problem", *Naval Research Logistics Quarterly* 18 (1971) 295–305.

[28] Yadegar, J., "Studies in combinatorial optimisation", PhD Thesis, University of London, 1984.